Proceedings of the 2001 IEEE/RSJ
International Conference on Intelligent Robots and Systems
Maui, Hawaii, USA, Oct. 29 - Nov. 03, 2001

Representation and Execution of Plan Sequences for Multi-Agent Systems

Paolo Pirjanian, Terry L. Huntsberger, Anthony Barrett
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, M/S 82-105
Pasadena, CA 91109-8099
{Paolo.Pirjanian}@jpl.nasa.gov

Abstract— Integration of planning and execution for single-agent systems has received considerable attention and numerous interesting approaches have been proposed. This is not the case for multi-agent systems, however. In this paper, we describe one approach to representing joint team activities using a Finite State Machine augmented with synchronization primitives for orchestration of group activities. This representation encodes a sequence of activities that can be distributed across a team of robots and executed in a coordinated manner. We then show how such a representation can be mapped onto a behavior-based formalism for execution, as supported by a distributed, multi-robot control architecture, CAMPOUT. We demonstrate the proposed mechanisms within the context of a challenging task, where two rovers collectively carry an extended container over rough terrain.

Index Terms — Team sequencing, tight coordination, distributed control architecture,

1 Introduction

NASA's enterprises are planning numerous high-priority missions, which involve fleets of highly autonomous cooperating agents ranging from constellations of satellites in Earth orbit to robotic communities for planetary outposts. For instance, *The Magnetotail Constellation DRACO* (Dynamics, Reconnection, And Configuration Observatory) is the Solar Terrestrial Probe (STP) mission [8] will use about 100 nanosatellites in order to generate the first global time-evolving maps of the fields and flows in the magnetosphere. Other proposed NASA missions plan to use groups of miniature, instrumented rovers of mass on the order of hundreds of grams [11]. Such nanorovers permit mobility-based science surveys on planetary surfaces with a small fraction of the science payload expected for currently planned, and future, rover missions.

All authors are with the Jet Propulsion Laboratory, California Institute of Technology. (address all correspondence to Paolo Pirjanian, telephone: 818-354-3169, e-mail: Paolo.Pirjanian@jpl.nasa.gov).

With a constellation of as many agents as is needed for such missions and the limited ground station coverage/capability (for technical & cost reasons) an autonomy function is vital. A challenge is thus to coordinate the activities of the agents in a coherent manner and subject to constraints imposed by the task, system, and the environment.

Most work in multi-robot coordination has to date been limited to tasks such as collective estimation [1, 2] (e.g., mapping and localization) cooperative foraging [3], cooperative box pushing [4] etc, where tight coordination of the activities of the robots is not required. Tightly-coupled coordination tasks are characterized by constraints imposed on the activities of one robot as a function of the state (e.g., position, velocity, etc.) of others and require both spatial and temporal coordination of activities of the team. Collective estimation and foraging tasks can be performed independently by each robot and do not require a tight coordination of team activities beyond possible task division. Cooperative box pushing requires tighter cooperation but can be accomplished with turn-taking schemes where each robot can alternate in pushing one end of the box towards a goal. But the spatial distribution of the robots and the timing of their activities are not critical since the box rests on a surface. Formation keeping tasks require even tighter coordination both spatially and temporally. But the strictest tasks in terms of coordination are those where tight temporal and spatial coordination is vital. For example, cooperative mobile object grasping, manipulation, and handling [5-7] (e.g., lifting and carrying, not pushing, a piano up the stairs) requires matriculate timing and precise spatial coordination of each robot in order to maintain grip of the object while manipulating or handling it.

We have developed a Control Architecture for Multi-robot Planetary OUTpost (CAMPOUT) which provides mechanisms for a range of coordination mechanisms suitable for both loosely coupled tasks as well as for tightly coupled tasks. In our treatment of the architecture, we emphasize the plan representation and execution aspects and discuss an approach for distributed planning and

0-7803-6612-3/01/$10.00©2001 IEEE

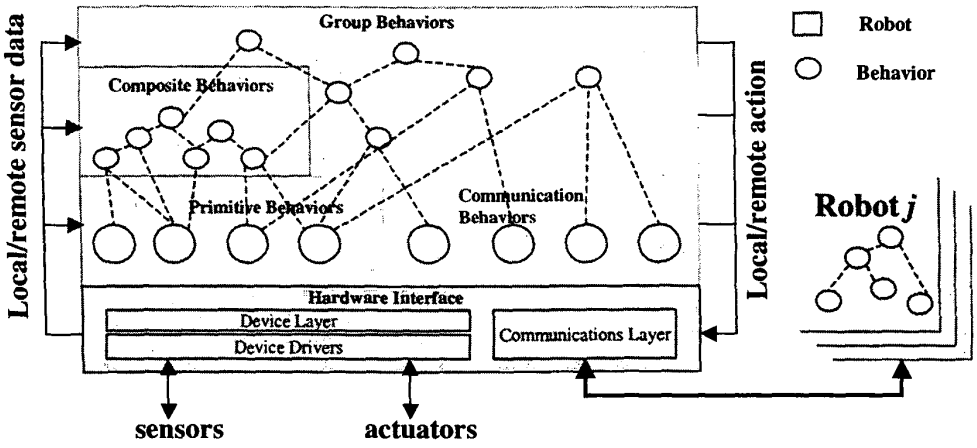

Figure 1 Schematic overview of CAMPOUT and its hierarchical organization in terms of primitive behaviors, composite behaviors built from primitive behaviors and group behaviors that are composed from coordination of behaviors across multiple robots. Each robot runs an instance of this architecture and coordinates activities through group behaviors, which is facilitated through the communications behaviors.

execution in the context of a task of cooperative object transportation using two rovers.

2 CAMPOUT

In a nutshell, CAMPOUT is a distributed control architecture based on a multi-agent or behavior-based methodology, wherein coordinated and cooperative group activities arise from coordination of behaviors across a team of robots/agents. Higher-level functionality is composed by coordination of more basic behaviors under the downward task decomposition of a multi-agent planner (see figure 1). In its current implementation, task decomposition is done by hand and encoded in as a distributed script/plan, which is then executed by the agents in coordination. We will describe how plans are devised, represented, and executed by CAMPOUT and also describe how we can automate the task of plan generation in distributed multi-robot systems.

Robotics is a highly multidisciplinary field, and requires efficient integration of many components (perception, mapping, localization, control, learning, etc.) that use different representations, frameworks, and paradigms (classical control theory, AI planners, estimation theory, data fusion, computer vision, utility theory, decision theory, fuzzy logic, multiple objective decision making etc.). CAMPOUT provides the infrastructure, tools, and guidelines that consolidate a number of diverse techniques to allow the efficient use and integration of these components for meaningful interaction and operation. This is facilitated through a few elementary architectural mechanisms for ***behavior representation***, ***behavior composition***, and ***team coordination*** and the interfaces between these. CAMPOUT is thus extensible and scales freely with regard to behavioral mechanisms and protocols it can host and fuse, re-mappable inter-robot communications it can support, and the overall ability to functionally integrate heterogeneous, multi-purpose platforms.

2.1 Behavior representation

In our architectural methodology we formalize a behavior, b, as a mapping, b: $P^* \times X \rightarrow [0; 1]$, that relates each percept sequence $p \in P^*$ and action $x \in X$ pair, (p, x), to a preference value that reflects the action's desirability. The percept describes possible (processed or raw) sensory input and the N-dimensional action space is defined to be a finite set of alternative actions. The described mapping assigns to each action $x \in X$ a preference, where the most desired actions are assigned 1 and undesired actions are assigned 0, from that behaviors point of view. Note that this definition of a behavior does not dictate how the mapping is to be implemented but provides a general recipe for a behavior with a well-defined interface (useful when composing behaviors regardless of their roles in a behavior hierarchy). This representation does not exclude implementation using a look-up-table, a finite state machine, a neural network, an expert system, control laws (such as PID etc.), or any other approach for that matter. Note also that this representation does not restrict us to reactive behaviors since it could have internal state. In that sense, each behavior can be implemented using whichever approach is appropriate. Finally, traditional, single-valued behaviors fall within this representation because, b: P^*

$\rightarrow X$ can be represented by a multi-valued output where all x are associated with 0 but the single x which is selected by the behavior.

2.2 Behavior composition

Behavior composition refers to the mechanism used for building higher-level behaviors by combining lower-level ones. This is achieved through the coordination of the activities of lower-level behaviors within the context of a high-level behavior's task and objective. An explicit design goal of CAMPOUT has been to support not one but an arbitrary number of Behavior Coordination Mechanisms (BCMs). In fact, the architecture can be extended by incorporation of new behavior coordination mechanisms. Since different BCMs often require different behavior representations, CAMPOUT uses a multi-valued behavior representation, which is general enough for a large class of applications. BCMs can be divided into two main classes: arbitration and command. CAMPOUT supports both classes. For a detailed overview, discussion, and comparison of behavior coordination mechanisms see [8].

2.3 Group coordination

In order to cooperate and collectively contribute to a common task the robots must cooperate and coordinate their activities. The coordination of a group of robots seems to have many similarities to behavior coordination within a single robot. The overarching idea for group coordination in CAMPOUT is to formulate the problem as *the coordination of multiple distributed behaviors, across a network of robots*, where more than one decision maker is present. Behavior coordination is basically concerned with resolving or managing conflicts between mutually exclusive alternatives and between behavioral objectives. This is, meanwhile, true for individual as well as group decision making. In this sense the difference between individual and group decision making is inessential and both can be studied in the same framework. Behavior coordination mechanisms that are typically used for coordination of behaviors of one robot can then be used for coordination of behaviors running on a network of robots. This way control loops can be produced that use sensors on one robot to drive a different robot. This mechanism is very powerful and the fundamental technique used in CAMPOUT for generating group behaviors (see figure 1).

Behavior coordination in multi-robot systems has received relatively little attention. One approach proposed in [9] uses inhibition and suppression across a network of heterogeneous robots augmented with motivational behaviors that can trigger behavior invocation based on some internal parameters that measure progress. A similar approach was proposed in the ALLYU architecture [10], which uses port arbitration as the main mechanism for multi-robot behavior coordination. Both these approaches can be viewed as the extension of subsumptive-style arbitration to multi-robot coordination. Recently, work in progress is investigating the extension of the 3T architecture [7] to multi-robot coordination.

The above approaches as well as most multi-robot architectures invariably have two things in common. First, multi-robot coordination mechanisms are limited to only

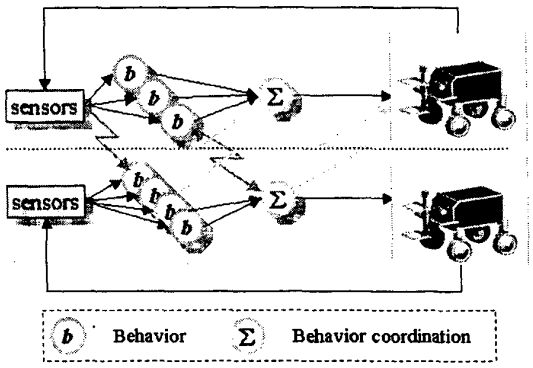

Figure 2 Networked robotics and resource sharing elements of CAMPOUT that enable definition of group coordination behaviors.

one approach and second which mostly tend to be based on arbitration rather than a command fusion schemes. We maintain that arbitration and command fusion mechanisms are complementary and a system implementation will typically make use of both. Command fusion mechanisms are inherently used for coordination of simultaneous activities and hence are more suitable for tightly-coupled tasks than arbitration mechanisms.

CAMPOUT can support an arbitrary number of behavior coordination mechanisms suitable for specific tasks. We have chosen to support, but not limit the architecture to, arbitration using ALLIANCE and ALLYU's subsumptive-style and behavior sequencing using discrete event systems (a finite-state-machine mechanism). Additionally, multi-objective behavior coordination is supported by CAMPOUT for command fusion. As we will show, we use the finite-state-machine formalism (regular expressions) as a representation for plans and group activities.

In CAMPOUT, multi-robot cooperation arises from coordination of multiple behaviors that reside on not one but a group of robots (see figure 2). To support this view, behavior coordination mechanisms should be extended to

Figure 3 Transport of an extended object requiring the tightly coupled co- ordination of multiple robot. (left) Column formation for long traverse. (right) Row formation for precision placement.

support multi-robot coordination. The behaviors and hence the robots can communicate implicitly by interaction through the environment or explicitly using sensory feedback or explicit communication. The first two approaches, interaction through the environment and sensory feedback, do not require any explicit form for architectural support as long as the robots have the necessary sensing capabilities to facilitate such interaction. These forms for interaction can be difficult and often computationally demanding, that is why most multi-robot systems resort to a form of explicit communication. CAMPOUT provides a rich and efficient infrastructure for explicit communication to facilitate multi-robot cooperation. Using this infrastructure, behaviors on one robot can interact with behaviors on other robots. In general the infrastructure defines a network of resources that can be shared among the robots. These resources include behaviors, sensors, and actuators. Thus a behavior on one robot can be driven by a sensor on another robot or even contribute to the control of a different robot. This idea is depicted in figure 2, where behavior composition can be achieved across several robots.

3 Plan representation and execution

In this section, we describe how the architectural components of CAMPOUT can be used to describe plans that represent group activities so that they can be executed by a distributed set of robots. We use a challenging task of collective object handling/transportation to describe these planning capabilities. Briefly, the task is to have two rovers carry an extended (2.5 m) container from a 'landing site' to a deployment area (see figure 3). Such an extended container would be difficult, if not impossible, to deploy using a solitary robot.

We have retrofitted two of our Sample Return Rovers (SRR&SRR2K) with a gimbal mounted on a cross-brace between the shoulders. The gimbal is not actuated but is fully instrumented with 6DOF force-torque sensors and pots and offers some mechanical compliance. The gimbal arrangement and two of the coordinated transport formations are shown in Figure 3. The coordinated transport task in open, uneven terrain requires a tightly-coupled, close coordination of the activities of the two robots. This is accomplished by some 20 behaviors, organized in a hierarchy within CAMPOUT. Here, we will only concentrate on a subset of those behaviors that are representative of the type of group behaviors that we can accomplish. In particular, we will focus on the description of a group behavior used for compliant formation keeping with the objective to assume a desired formation. The formation can be defined by the angle between the two rovers and the heading of the formation towards a target (see figure 4). For instance a row formation is formed when the two rovers are positioned side-by-side (see right picture in figure 3).

3.1 Compliant formation keeping

The compliant formation keeping group activity is invoked to configure the two robots into a given formation, defined by the relative angle between them, α, and the relative angle towards the target, γ (see figure 4). A *Face Target* behavior provides the angle to the target then the *Turn* group behavior reconfigures the formation to a desired one. Two constraints make this a challenging task.

First, transformation between the current and target formations must ensure that the container is handled safely, i.e., the distance between the robots, *d*, should always remain within some tolerance margin, $d_{min} \leq d \leq d_{max}$, determined by the distance between the grip points of the rovers, L (200 cm), and the longitudinal translation in the gimbal T_{gimbal} (±2 cm). I.e., $L-2T_{gimbal} \leq d \leq L+2T_{gimbal}$, which implies that the distance between the two rovers should be maintained within a margin of 8cm ($4T_{gimbal}$). A set of compliance behaviors, described later, monitor the state of the load and constrain the movement of the rovers to guarantee this requirement.

Second, it is required that the container does not collide with the mast on the lead rover (see figures 3 and 4), which could lead to damaging the mast, the gripper/gimbal, or the container, and/or dropping the container. The shaded area around the lead rover indicates the safety zone (-35 to 35 degrees) where the container beam cannot enter because it will then collide with the mast. We used the following distributed strategy for accomplishing this task, with respect to the second constraint, and hand-coded this strategy as a group behavior:

1. The lead rover turns in place as far as possible until either θ_{target} is reached or it cannot move further due to the safety zone constraint.
2. The follower pivots around the lead rover until either $\alpha_{formation}$ is reached or it cannot move further due to the safety zone constraint.

This sequence will alternate the two until the goal configuration is reached. Note that once one rover moves it also frees the other rover from being constrained by the safety zone. In this way, incremental progress is made towards the goal configuration while respecting the second constraint imposed by the safety zone. This strategy is encoded as a Distributed Finite State Machine (DFSM) (see figure 5), which is basically a Finite State Machine augmented with communication links. Each state represents the activation of a (set of) behavior(s) and transitions between the behaviors are triggered by events as indicated on the arrows. These events can be generated either by perceptual feedback or explicit communication between robots. Note on figure 5 that the DFSM representation has two parts: one that runs on rover 1 and a replicate that runs on rover 2. Note also that a finite state machine is a behavior coordination mechanism that is supported by CAMPOUT for behavior arbitration. The states, represented by bubbles in the figure, correspond to primitive or composite behaviors implemented within CAMPOUT.

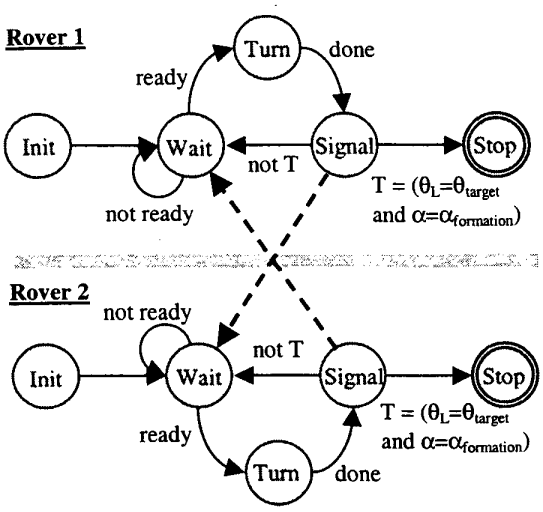

Figure 5 Distributed plan used for the compliant formation-keeping task. The arrows represent events that cause transitions and the dashed curves represent events caused by explicit communication of signals.

Revisiting the formation keeping strategy, we see in the figure that one rover turns (using Turn behavior) until it is done (i.e., cannot turn further) then hands the token to the other rover by a signal and waits. The *Turn* behavior has different implementation for each of the rovers; for the follower it consists of Ackerman turns and for the leader it consists of turn-in-place. The *Wait* behavior in each of the rovers consists of a number of behaviors including compliance behaviors. I.e., when the other rover starts moving/turning the waiting rover monitors the state of the load (through the sensors of the gimbal) and then triggers a compliance behavior to assure that the container is handled safely in accordance with the distance constraint describe above. This is accomplished by crabbing in the direction of the container in order to center the load (based on pot-meter readings) and to reduce the forces on the gimbals (based on the force-torque sensor readings).

Using this strategy, our experimental studies demonstrated:

- 40-to-50 meter autonomous traverses of outdoor irregular terrain (maximal slope of 9°) by two rovers (SRR/SRR2k) in the tightly coupled transport of an extended container,
- autonomous change of formation by two rovers carrying an extended container under compliant control, and
- continuous visual guidance to a designated deployment site from 50 m, heading error < 1°; distance error < 5% at 8m by use of a visual template.

More importantly, we demonstrated how group activities can be encoded using the underlying mechanisms

provided by CAMPOUT and encoded essentially as a distributed finite state machine (which itself is a behavior coordination mechanism) across a group of robots. As described in the next section, we are currently investigating robust ways to string multiple group activities together.

4 Distributed Team Sequencing

While the distributed finite state automaton approach benefits from conceptual simplicity, manually crafting such automata becomes a tedious error-prone task as the size of the group behaviors increases. Unfortunately experience within the multi-agent community has shown that significant numbers of unanticipated interactions between agents (like rovers) show up as people attempt to manually engineer interacting controllers between more than 3 agents [11]. These interactions lead to activity termination at best and mission termination at worst.

We can apply teamwork models [11, 12] to reduce the complexity problem by giving the rovers a shared team-state. Here each rover can monitor its own performance and selectively transmit results to its teammates. Partitioning the system's state into local rover states and shared team-states facilitates this selective transmission. While the rovers keep their local states private, they communicate to keep team-states consistent across teams in the remote outpost.

4.1 Representing team sequences

In addition to regular behaviors found in the original approach, team sequences also include team behaviors. These define coordination points where the team synchronizes before and after executing the team activity consisting of group and other behaviors. For instance, a team behavior to control our 2-rover transportation activity might have 5 behaviors to coordinate the rovers while as they approach their payload, pick it up, transport it, assume formation, and put it down. Just like group and composite behaviors, team activities are defined in terms of more primitive team activities as well as behaviors.

From a representational standpoint, team activities are similar to group behaviors. The only additions to turn a group behavior into a team activity involve defining teams to perform sub-behaviors and assigning roles to teammates. More precisely, injecting a model of teamwork into an existing hierarchical system, like CAMPOUT, involves adding three features:

- generalization of group behaviors to represent team activities with role assignments,
- representation of team and/or sub-team states, and
- restrictions to only let a teammate modify a team state through a team activity.

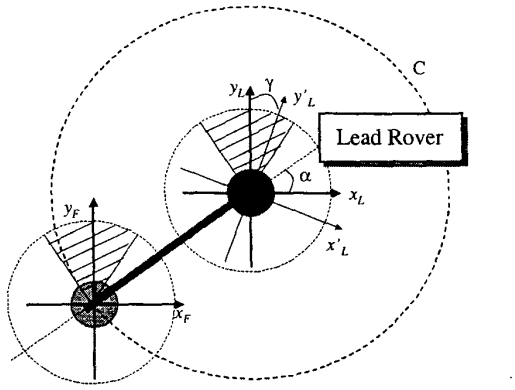

Figure 4 Formation between the two robots with follower on left and leader on right. The formation is defined by the angle α between the two robots. Desired heading is given by the relative heading angle γ. The shaded area on the lead rover is a safety zone where the container beam should not enter to prevent collisions with its mast.

Instead of having each rover follow a separate sequence of composite and group behaviors, the rovers follow roles in a single team activity. This lets each rover actively monitor its own progress and passively track its teammates' performance. This passive monitoring process maintains robustness by facilitating the creation of general-purpose error detection techniques based on shared team-state.

While this example's impoverished number of rovers does not sufficiently motivate the need for teamwork, other mission proposals describe over a dozen, or even a hundred, rovers to support a robotic outpost. To support teamwork for these larger missions, we must alter the underlying behavioral architectures to manage each rover team's associated team-state. We illuminate these changes by describing the machinery underlying team activity execution.

4.2 Performing team behaviors

A team of rovers contains a leader and one or more followers that jointly intend to accomplish some task by executing a team activity. Teams dynamically form when team activity execution starts and dissolve upon completion. When a team performs a task, it shares a team-state. This state contains facts like a list of teammates, their roles in performing the joint task, and other information to coordinate team activity.

Depending on the team activity, execution manipulates the behaviors to alter parts of the local and team-state

information. Since team-states are replicated across all teammates, a rover must broadcast all team-state changes to maintain consistency. The standard protocol for changing a team-state is a 3-step process where one rover broadcasts the change, all teammates broadcast acknowledgements in turn, and all teammates update their copies upon hearing everyone else. If a teammate does not respond before a time-out interval, the original *rover* rebroadcasts the change.

While only transmitting team-state changes reduce communications, the number of broadcasts still implies bandwidth problems as the rover population increases. Stopping a rover from broadcasting a change when teammates can infer it from observation further reduces communications [11, 13]. For instance, the rovers in our transportation example do not have to signal the end of a pickup activity. The mere act of sensing that the bar is seated properly in the holder tells the rovers that the pickup activity is over.

5 Conclusions

In this paper, we described CAMPOUT and its mechanisms for representation and execution of joint team activities. In particular, we have demonstrated that simple group activities can be encoded as a distributed finite state machine (DFSM), which is basically an extension of a FSM arbitration mechanism to team coordination. While DFSMs provide a conceptually simple yet formally powerful tool for representing team plans, it can be tedious to hand code them for complex tasks due to a combinatorial explosion in number of states. We argued that using a model of teamwork and dynamic task allocation mechanism can enable a general and powerful solution to team plan generation and provide protocols for inter-agent interaction to support reliable execution.

We are currently pursuing this line of research for tasks involving multiple satellites for space science missions.

6 Acknowledgements

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

The authors would like to thank the entire team in the Robotics and Mechanical Engineering group at JPL for doing a great job in mechanical construction, controls, implementation, and field-testing. In particular, we would like to thank Paul S. Schenker, Ashitey Trebi-Ollennu, Hari Das, Hrand Aghazarian, Anthony Ganino, Mike Garrett, and Sanjay S. Joshi.

References

[1] S. I. Roumeliotis and G.A. Bekey, "Collective Localization: A distributed Kalman filter approach to localization of groups of mobile robots". In Proc. 2000 IEEE International Conference on Robotics and Automation, San Francisco, CA, April 22-28, pp. 2958-2965.

[2] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A Probabilistic Approach to Collaborative Multi-Robot Localization.", Auonomous Robots, 8, (3), 2000.

[3] M.J Mataric, "Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior", Journal of Experimental and Theoretical Artificial Intelligence, special issue on Software Architectures for Physical Agents, 9(2-3), H. Hexmoor, I. Horswill, and D. Kortenkamp, eds., 1997, 323-336.

[4] D. Rus, B. Donald, and J. Jennings, "Moving furniture with teams of autonomoujs robots." In Proceedins of IEEE/RSJ International Conference on Intelligent Robots and Systems, pges 235-2442, 1995.

[5] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holberg, and A. Casal, "Decentralized cooperation between multiple manipulators," 5th IEEE Int. Workshop on Robot and Human Communication, 1996;

[6] M. Hara, M. Fukuda, H. Nishibashi, Y. Aiyama, J. Ota, and T. Arai, "Motion control of cooperative transportation system by quadruped robots based on vibration model walking," IROS'99

[7] R. Simmons, S. Singh, D. Hershberger, J. Ramos, T. Smith. "First Results in the Coordination of Heterogeneous Robots for Large-Scale Assembly", In Proceedings of the International Symposium on Experimental Robotics (ISER), Honolulu Hawaii, December 2000.

[8] P. Pirjanian, "Behavior Coordination Mechanisms - State-of-the-art", Tech-report IRIS-99-375, Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California, October, 1999.

[9] L.E. Parker, "Heterogeneous Multi-Robot Cooperation", Massachusetts Institute of Technology Ph.D. Dissertation, January 1994. Available as MIT Artificial Intelligence Laboratory Technical Report 1465, February 1994.

[10] B.B. Werger, "Ayllu: Distributed Port-Arbitrated Behavior-Based Control," in Distributed Autonomous Robotic Systems 4, Lynne E. Parker, George Bekey, and Jacob Barhen (eds.), Springer, 2000:25-34.

[11] M. Tambe, "Towards Flexible Teamwork," *Journal of Artificial Intelligence Research*, 7:83-124.

[12] P. Stone and M. Veloso, "Task Decomposition and Dynamic Role Assignment for Real-Time Strategic Teamwork," Submitted to ICMAS'98.

[13] M. Huber and E. Durfee, "On Acting Together: Without Communication," AAAI Spring Symp-osium on Representing Mental States and Mechanisms, 1995.